Problem Definition:
The given problem ask us to classify the given video depending upon the 101 actions classes. For this project I am considering only first 15 classes.

Preprocessing:
Features from the images in the data are created using VGGRelu16.
The data given is of the form each video divided into 25 frames.
As input of VGG needs to be of size (244,244) each of the 25 frame needs to be resized. Instead of completely resizing the image to this given shape it is better to use FiveCrop function from the torch.transforms which crops 5 (224, 224) sized parts of the given image from its center, and four corners. This saves the semantic information of the image (actions objects) from getting randomly cropped and thus results into better features. These five images are stacked together to form one single frame. Further normalization of each of the these cropped image brings them in fixed range. The formula used for this is:

```
for each channel in our case 3(R, G, B) do:
        (input[channel] =(input[channel] - mean[channel]) / std[channel])
where,
        mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]
```

To sum up at the end of the preprocessing step is 5 images for each of the 25 images of 1 video. These are stored in a dictionary with key as photo_id(dirname). For the next step I send all these as input to the VGG.

Feature Extraction:

Features are extracted per image using pre trained VGG model of pytorch by freezing all layers of the model and passing each image through the model and taking features from the second last layer. These are of the size 4096. The input of vgg is of size (5, 3, 244, 244) 5 being 5 crops described above. Mean of the output of vgg is taken to convert 5 img crop to one feature. thus giving feature of size(1, 4096). Thus feature of each video is found by stacking all 25 image features of size (1, 4096) thus leading to final size of (25, 4096). VGG in general perform better than most of newer models for video classification because the don't downsample very fast maintaining the action and object information(global) information needed for the action classification problem.
https://www.reddit.com/r/MachineLearning/comments/7rrrk3/d_eat_your_vggtables_or_why_does_neural_style/

Action Classification:
In this part of the assignment, features of vgg are used for action classification. As each of the 25 frames are part of one video where action can get divided across the frames, it is important to store the information of the previous frame while making a decision for the current frame. Therefor the best model for doing this is LSTM:

The first model I tried was a simple LSTM where each frame of VGG output forms input of the single unit of LSTM model. Each units are interconnected. Giving current unit information learned in the previous LSTM unit. Thus input of this model is (Batchsize, no. of Frames, Featuresize) the lstm structure is (featuresize, no. of frames, layers) layers are the numbers of lstm layers stacked above one another. The output of the last Lstm unit is output, hidden state information of last unit and channel state information of last unit. The first one of this is sent to linear layer which acts as the classification layer.
This model got to accuracy of **80% at 21 epochs.**
I first tried with SGD optimizer and cross entropy loss but the accuracy increased very slowly and the loss convergence took a very long time. eg

```
Epoch 43, Train Accuracy: 24.410540915395284, Train Loss: .10522033090234299
Starting epoch number 44
Epoch 44, Train Accuracy: 24.895977808599167, Train Loss: .10505214610476765
```

So I changed my optimizer to Adam optimizer
I made use of small set of validation to find breakpoint. At this point the validation loss jumped up like this:

```
Epoch 20, Validation Accuracy: 93.79310344827586, Validation Loss: .1283331988700505
Starting epoch number 21
Epoch 21, Train Accuracy: 95.83654587509638, Train Loss: .05687454565296792
Epoch 21, Validation Accuracy: 84.82758620689656, Validation Loss: .2753363991605824
```

The other model that I tried was with a intuition that the information within all frames should be modelled together in a linear layer and then sent to the LSTM's like done with most of the sentences in english. But as images are with 4096 dimensions sending them directly to linear layer is too many parameters to learn for the fully connected network. So I first reduce the shape of the image to smaller and smaller using convolutional kernels and maxpooling(2 convolutions and two maxpooling) with interleaving of normalization and activation function and dropout for maintaining regularization.
The input of the convolution layer is (batchsize, 1, no. of frames, features) I added one in place of channels to perform 2d convolution. The output of this is then flattened and sent to the linear fully connected layer. The output of this layer then becomes input of the LSTM. Input of the LSTM is now of size (batchsize, 1, no. of output nodes of linear layer) notice this successfully combined the 25 frames to one. The output of this layer is feed to linear layer for classification the output nodes in this layer are 15 which are equal to the number of classes.

The adam optimizer in this model gave wavering loss which was not decreasing like this:

```
Epoch 1, Train Accuracy: 65.53585196607556, Train Loss: 1.0981400654512639
Epoch 1, Validation Accuracy: 75.86206896551724, Validation Loss: .9634323284543795
Starting epoch number 2
Epoch 2, Train Accuracy: 6.553585196607556, Train Loss: 1.0985073348422555
Epoch 2, Validation Accuracy: 75.86206896551724, Validation Loss: .9672307639286435
Starting epoch number 3
Epoch 3, Train Accuracy: 64.764841942945255, Train Loss: 1.0988101742317241
Epoch 3, Validation Accuracy: 41.37931034482759, Validation Loss: .9696672867084372
Starting epoch number 4
Epoch 4, Train Accuracy: 6.399383191981496, Train Loss: 1.0986555091399088
Epoch 4, Validation Accuracy: 6.206896551724138, Validation Loss: .9657204233366867
```

I hence tried the optimizer to AdaDelta optimizer which gave better result. I trained this final model for 47 epochs the loss was still decreasing but very slowly still. The accuracy I got at this point with my model was **87.4545 with 47 epochs (final model)**

The other model I tried was instead of using my own convolutions I tried calling the InceptionB function of torch which has inbuilt convolutions but the Gpu memory went out with this model. I didn't try it on my local machine.

Evaluation:
Evaluation of the model on test set was done with keeping the above described models in eval mode and passing the test set with the data loader. The other technique used for evaluation was svm. Here all the features from all the frames in the video were combined together to a single feature by stacking and then passed to LinearSVC resulting in accuracy of 90.14%.